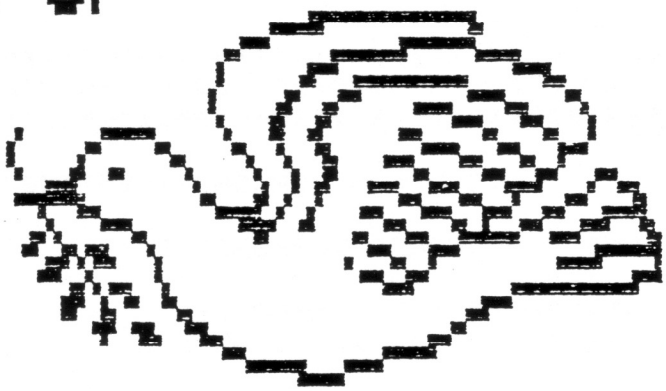
VZ-LINK MAGAZINE

NEW ZEALAND

VZ 300/200

COMPUTER USERS CLUB

ISSUE 37



NOVEMBER 89

COMMENT

We are attempting to make this Newsletter as lively and as interesting as possible.
In order to do this we once again ask for your contributions.

I would like to welcome another new member,Steve Dale from Devonport.Steve sent in a program with his Sub's and it will appear next issue.Thanks Steve.

This club has been going for 3 years now,Doesn't time fly ?.
I hope you are getting what you require from the club.
Remember,I have Heap,s of other information and Programs on paper if you need it.
Maybe what you seek is Available and Just a phone call or visit away.

This issue see's the last part of the VZ DOS item which I forgot to include last issue.SORRY.

We also have the continuation of Part 1 of Micro Magic.

There is also a letter from Lou Ellmers,Lou is having problems.
Maybe you can help.Tell us all,if you have the answers.

Don't forget the last club meeting for 89.
on Saturday 2nd December.
(Details below)

See you there.

Peter

****
AUCKLAND VZ 300/200 COMPUTER USERS CLUB
****

VZ USERS-WORKSHOP/MEETING

WE WILL BE HOLDING A VZ USERS WORKSHOP ON:-

SATURDAY 2 DECEMBER 89
STARTING AT 10-30 AM

AT THE :-

OWAIRAKA BAPTIST CHURCH
119 RICHARDSON ROAD
OWAIRAKA

BRING YOUR FAVOURITE SOFTWARE ETC
BRING YOUR PROBLEM'S

A LIGHT LUNCH WILL BE PROVIDED,BUT PLEASE BRING A PLATE TO HELP

MICRO MAGIC

Beginner's Guide to the Veezed ~

You may have read that Programs should be STRUCTURED. Well, BUTCHER was not structured in the manner that many Programmers would recommend. When writing a Program from scratch it is usually necessary to draw up a FLOW CHART. A FLOW CHART is something that looks like a tree. The main stem has a lot of GOSUB branches. To design a Program one draws a vertical line to represent the Program from the first line to the end. This line would be broken by inserting boxes in which you write words such as "TITLE", "INPUT (WORD", "INPUT (CLUE)" and so on. Branches leading to other boxes would then radiate from the main stem boxes where sub-routines are noted with such words as "FIND LENGTH OF WORD", "PUT IN ERROR TRAP", and anything else that is needed before RETURNING to the main stem again. You will probably use BASIC as a form of shorthand when drawing up a flow chart. Once the FLOW CHART look s as though you've covered everything, you can then add consecutive numbers to the boxes on the main stem, and larger block numbers to all the sub-routine boxes. The idea of keeping the sub-routine line numbers higher than the main stem is to help speed up the running of the program. Any sub-routine which is not used more than once would have the highest block of line numbers. You see, every time a LOOP cycles, such as in BUTCHER, the computer scans every line from the lowest line number to the next one to be processed. Programs with heaps of DATA always need to have the DATA at the bottom of the program out of the way of LOOPS and GOSUBs. But, more of this later. MICRO MAGIC will be Publishing Programs where this can be demonstrated. In the mean time, you might try drawing up a FLOW CHART for BUTCHER. As a clue you should have the ERROR TRAP in line 140 as a sub-routine, which means the GOTO120 should still remain in the main stem of the Program.

****

When we compare the GUESSLETTER with the first and subsequent letters of the WORD, this is possible because the computer only deals in numbers. Every letter has what is called an ASCII NUMBER. If you were to type PRINTASC ("B") the computer would Print 66. Later we see how by changing the ASCII numbers we can make a printer or plotter print lower-case letters. We can also compare letters and have them arranged in alphabetical order.

P.O.BOX 1972 C.P.O AUCKLAND N.Z

(C) AUCKLAND VZ 300/200 COMPUTER USER'S CLUB 1986

ALL MATERIAL IN THIS NEWSLETTER IS SUBJECT TO COPYRIGHT..
CONTRIBUTED PROGRAMS & ARTICLES CAN BE REPRODUCED ON THE UNDERSTANDING THAT THEY ARE FOR THE PRIVATE USE OF SUBSCRIBERS ONLY.
COPYRIGHT IS ALWAY'S RETAINED BY THE AUTHOR.
THIS CLUB WILL NOT BE HELD RESPONSIBLE FOR THE BREECHING OF COPYRIGHT CAUSED BY THE PUBLICATION OF ANY MATERIAL SENT IN UNDER A FALSE NAME.

YOU ARE WELCOME TO PHONE ME IF YOU HAVE PROBLEMS
PHONE 761-451

BUTCHER

```
10 ' HANG THE BUTCHER "MICRO
20 '  GORDON BROWELL  "MAGIC
30 'SLIGHTLY MODIFIED "NOV 8
100 CLS:CLEAR:COLOR3,1:GOSUB570
110 PRINT"DO NOT LET PLAYER SEE THE WORD ":SOUND28,1
120 COLOR7:PRINT@195,"ENTER WORD";:INPUTA$
130 PRINT@206,"              ":SOUND28,3
140 IFLEN(A$)>10THENPRINT@234,"TOO LONG!":GOSUB570:GOTO120
150 PRINT@259,"ENTER CLUE";:INPUTZ$:CLS
160 A=LEN(A$):PRINT@289,"";:PRINTUSING"##";A;:PRINT"  LETTERS"
170 B=A+6:PRINT@384,B" GUESSES":PRINT@129,"CLUE:  "Z$
180 PRINT@161,"GUESS COUNT":PRINT@193,"ERROR COUNT"
190 FORX=6TOA*2+4STEP2:PRINT@X,"_":NEXT
200 FORI=1TOA:B$(I)=MID$(A$,I,1):NEXT
210 L=448
220 FORJ=1TOB:PRINT@70,"";:INPUT"GUESS LETTER";X$
230 PRINT@L,X$:L=L+2
240 P=6:FORI=1TOA
250 IFX$=B$(I)THENPRINT@P,X$
260 IFX$=B$(I)THENC=1
270 IFX$=B$(I)THENR=R+1
272 IFR=ATHEN350
280 P=P+2:NEXT
290 IFC=0THEND=D+1
300 IFC=1THENPRINT@225,"  RIGHT!  "ELSEPRINT@225,"  WRONG!  "
310 IFC=0THENGOSUB360
320 C=0:PRINT@204,D
330 PRINT@172,J
340 NEXT
342 CLS:PRINT@33,"SORRY!":PRINT" YOU'VE RUN OUT OF GUESSES."
343 GOTO600
350 CLS:PRINT@197,"YOU GOT IT!  "A$:GOTO560
360 IFD=1,390ELSEIFD=2,430ELSEIFD=3,440ELSEIFD=4,460
370 IFD=5,470ELSEIFD=6,480ELSEIFD=7,490ELSEIFD=8,500
380 IFD=9,510ELSEIFD=10,520
390 COLOR7:PRINT@113,"[redacted]"
400 COLOR7:PRINT@113,"[redacted]"
410 FORI=157TO413STEP32:PRINT@I,"[redacted]":NEXT
420 PRINT@433,"[redacted]":RETURN
430 COLOR5:PRINT@148,"[redacted]":PRINT@180,"[redacted]":RETURN
440 COLOR5:PRINT@213,"[redacted]":COLOR8:PRINT@243,"[redacted]"
450 PRINT@276,"[redacted]":RETURN
460 COLOR8:PRINT@248,"[redacted]":COLOR5:PRINT@250,"[redacted]":RETURN
470 COLOR5:PRINT@239,"[redacted]":COLOR8:PRINT@241,"[redacted]":RETURN
480 PRINT@308,"[redacted]":PRINT@340,"   ":RETURN
490 PRINT@372," ":PRINT@404," ":RETURN
500 PRINT@374," ":PRINT@406," ":RETURN
510 COLOR4:PRINT@434,"[redacted]":RETURN
520 PRINT@433,"              "
530 PRINT@433,"              ":PRINT@434,"[redacted]":SOUND1,6
540 N=129:FORI=1TO16:PRINT@N," ":N=N+1:NEXT:PRINT@129,A$
550 PRINT@152,"EE-EEK!":SOUND31,9
560 FORI=1TO3000:NEXT:RUN
570 SOUND1,5:CLS:PRINT"[redacted]"
580 PRINT@32,"H A N G  T H E  B U T C H E R"
590 PRINT@64,"[redacted]":RETURN
600 GOSUB390:GOSUB430:GOSUB440:GOSUB460:GOSUB470:GOSUB480
610 GOSUB490:GOSUB500:GOSUB510:GOTO520
```

MICRO MAGIC WORKSHOP

There are so many printers on the market today that it would be impractical to try to mention them all. The best we can do is to select one that is typical, and the one I have chosen is the STAR SG-10.

A printer for use with the VZ-200/300 needs to be a Centronics Parallel type. Without getting technical that means the VeeZed Printer Interface will fit and will operate the printer.

We have already mentioned that the printer holds in memory the ASCII Code of numbers corresponding to the characters it will print. As the VeeZed does not have lower case characters and the printer does, then obviously the ASCII Code of the VeeZed is not the standard code. The standard code (from 0 to 127) is published in MICRO MAGIC. Can you devise a FOR..TO..NEXT loop that will print out a list of the computer's ASCII Code from 32 to 255? You could then compare the printable characters of the printer and the computer. This was how the sub-routine for the ASCII Conversion in MICRO TYPE was devised. Maybe, you can improve on this one.

The Dick Smith GP-100 uses blocks of ASCII numbers to simulate the graphics of the VeeZeds. It will LLIST a program that has graphics in the program lines. It will allow COPY to be used in both MODE(0) and MODE(1). Everything seems to be in favour of using the GP-100 with the VeeZed. Unfortunately, there are a couple of disadvantages. The first is that the actual type-face is not very atractive. The characters have no DESCENDERS. They are that part of a letter such as p, q or y that DESCEND below the print line. The other disadvantage is that Graphic characters take up more space than a text character making a mess of a COPY or screen dump of mixed graphics and text. This printer will not take plain paper (only tractor-feed) and uses an expensive ribbon cartridge that doesn't last too long. This will give you some idea of what to look for when buying a new printer.

The printer being used to prepare this page is a STAR SG-10. Any Printer Control Codes mentioned here are those appropriate to this printer; yours may be different, but they are typical. The Operating Manual for most printers has a list of Commands that will act as Control Codes. Most begin with ESC. As you're aware the VeeZed does not have an ESC (ESCAPE) key. We do not know or understand why the VeeZed does not have an AUTO, CLEAR or ESCAPE key; but that's another story.

ESC (ESCAPE) is CHR$(27). In a program where ESC is to be used many times, E$=CHR$(27) will appear at the beginning of a program. A typical Control Code may be given as ESC"B"CHR$(4). To send that to the printer you would type;

```
LPRINTCHR$(27)"B"CHR$(4);
```

or if E$=CHR$(27) has been included in a program, the command would be written in a sub-routine as;

```
LPRINTE$"B"CHR$(4);:RETURN
```

Note that there are NO COLONS or COMMAS in the command itself. The semi-colon at the end is to prevent the printer from doing an unwanted line-feed. An alternative to the letter in quotes would be to enter it as CHR$(66) which is a "B". If your list of Printer Commands looks like ESC "B" 2, the ESC is CHR$(27), the "B" can be "B" or CHR$(66) and the 2 CHR$(2). There are some commands that do not require ESC and are usually just the single CHR$(15), or similar. And, if your list shows a LOWER case letter, then the standard ASCII CODE table will provide the CHR$(...) number to use. There are lots of variations such as SI 15. A good rule is that if ESC is not specifically given then, in this example, CHR$(15) is all that is required.

PRINTER EXPERIMENT.

Here is a very much shortened version of MICRO TYPE. It will enable you to experiment with the PRINTER COMMAND of your particular printer

Studio Ad Lib

```
10 CLEAR3000:DIMZ$(20):E$=CHR$(27):L=1:CLS
20 PRINT@286,"":INPUTZ$(L):W=LEN(Z$(L)):CLS
30 FORJ=1TOL:FORI=1TOW
40 A=ASC(RIGHT$(LEFT$(Z$(J),I),1))
50 REM
60 REM
70 IFA=91THENA=0:GOSUB200        ' 91 IS SQUARE BRACKET SHIFTO
80 IFA=93THENA=0:GOSUB210        ' 93 IS SQUARE BRACKET SHIFTP
90 IFA>=64ANDA<=95THENA=A+32     ' FOR LOWER CASE
100 IFA>=192ANDA<=223THENA=A-128' FOR UPPER CASE
110 IFA>=224ANDA<=255THENA=A-192' FOR INVERSE TO NORMAL
120 LPRINTCHR$(A);:NEXT
130 LPRINT:PRINTZ$(J):NEXT
140 L=L+1:GOTO20
200 LPRINTCHR$(14);:RETURN        ' EXPANDED - ON
210 LPRINTCHR$(15);:RETURN        ' EXPANDED - OFF
```

GP-100 OWNERS When you RUN this, type a line of text putting a word in SQUARE BRACKETS. Thats those on keys O and P. That word will appear in EXPANDED type.
Then add these lines:

```
50 IFA=35THENA=0:GOSUB220    '35 is the # sign (SHIFT 3).
220 W=LEN(Z$(J)-1:LPRINTCHR$(20);
230 FORU=1TOW:LPRINTCHR$(95);:NEXT:RETURN
```

Put a HASH sign (SHIFT3) AFTER a few words in a line of text and the printer will UNDERLINE those words. CHR$(20) sends the print head back to the beginning of the line without a line-feed and the loop in line 230 will type the Underline back to where you put the SHIFT2. Now you know why the UNDERLINE is unsatisfactory on this printer. It overprints the bottom of the words. You may find a use for it.

OTHER PRINTERS Change lines 200 & 210 to Control Codes taken from your printer manual. Use practical ones such as *ITALICS* or CONDENSED. By putting a word in SQUARE BRACKETS (SHIFTO and SHITP) the printer will obey your Controls. The REMs will give you the opportunity of working out a system of your own.

CLUE If you typed PRINTASC(" ") the computer would return a number for whatever you put between the quotes. If, say, you put a SHIFTQ GRAPHIC then the printer would return 144. And if you did that again with a SHIFTW GRAPHIC you would get 151.
OK, supposing you made lines 50 and 60:

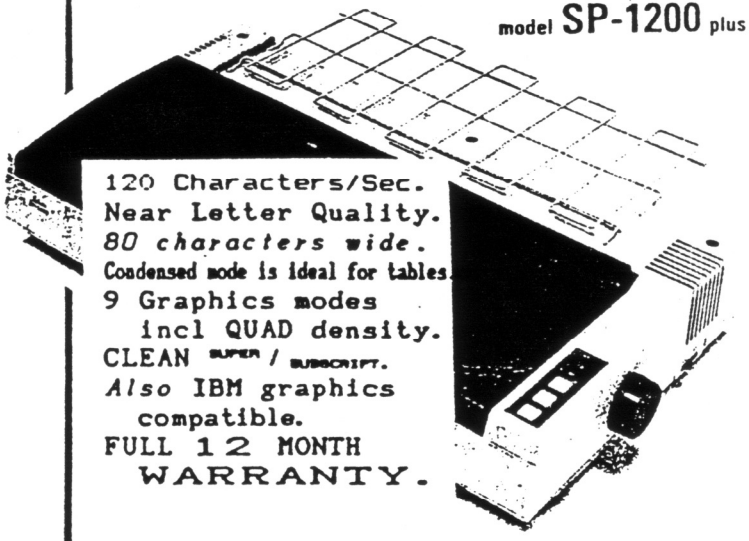
```
50 IFA=144THENA=0:GOSUB240
60 IFA=151THENA=0:GOSUB250
```

you could then access any control code in lines 240 and 250, using 240 to turn the control ON and 250 to turn it OFF again. To use this control in the text you only have to put a SHIFTQ in front of a word and SHIFTW after it. And it doesn't have to be just a word. It can be one letter, one word or a number of words. Don't forget that E$=CHR$(27). and don't forget the ;

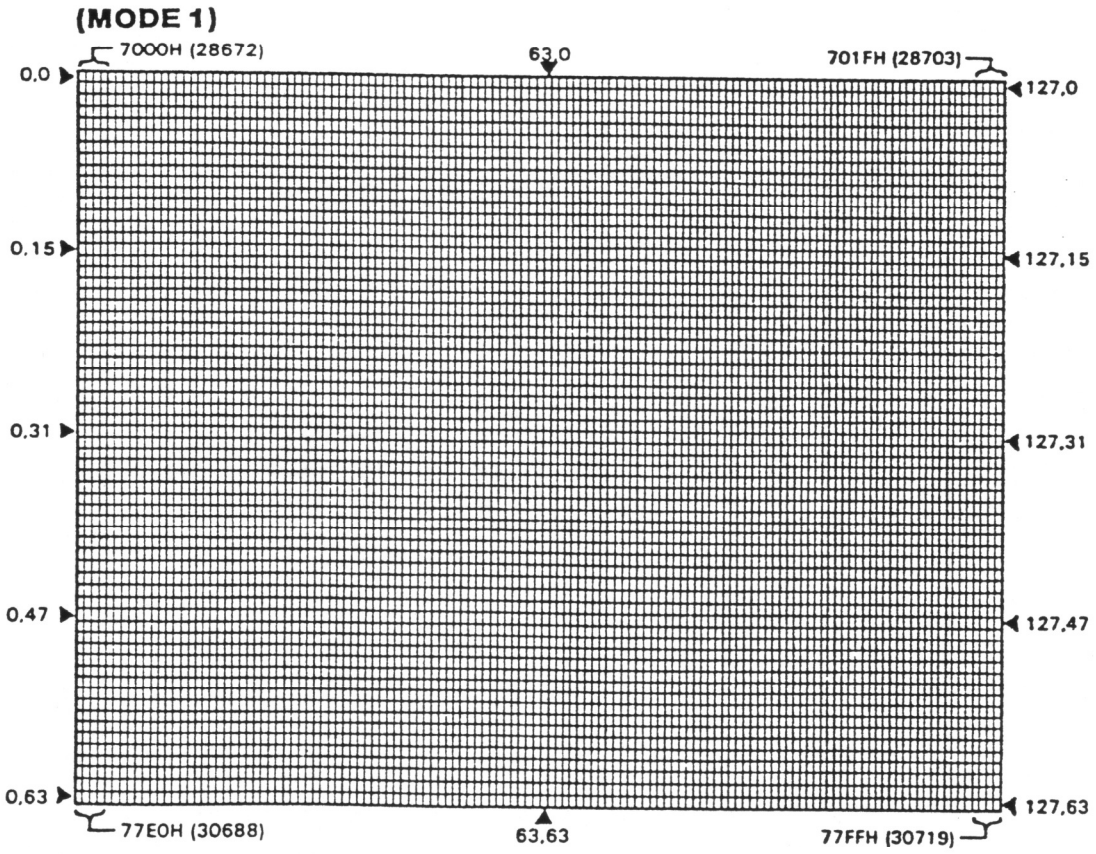
One last word on Control Codes for printers. Sometimes a code will take precedence over another. For instance, if I select NLQ, (Near Letter Quality), some of the other codes cease to work. While in NLQ I cannot use SUPERSCIPT, SUBSCRIPT and some others. Only by referring to your Operating Manual can you see what restrictions there are.

If you have any problems with your particular printer then drop me a line with a sample of the codes from your book and I will try to help.

```
[redacted] 1

SCROLL        EDIT

PRINT

FILE

RETRIEVE

CONTINUE

[redacted] 0

MICRO TYPE MENU
```



TIP.Consider taking your VeeZed and a Printer Interface with you when you go shopping for a printer and give some controls a try to be sure its going to work at home. Try a few LPRINTs in case the printer is dedicated to IBM. A reel-to-reel ribbon lasts longer and is cheaper and to be able to use plain paper, as well as fan-fold tractor feed, can be very handy. Friction feed will take envelopes also.



Although MICRO MAGIC is a beginner's guide to BASIC, there are users who are interested in Machine Code programming. The next Studio Ad Lib will contain an explanation of what Machine Code is. A couple of experimental programs will be given to help in discovering what parts of the computer's memory locations are used for. And for those already using POKES into memory there will be a table of TOKENS.

VZ DOS

In the last part in this series on the VZ DOS system, we looked at a routine to read a sector of the disk from BASIC using a short machine code program. Here are the programs as they stood last time:

ASSEMBLY LANGUAGE SOURCE CODE LISTING

```
001     PUSH AF
002     PUSH BC
003     PUSH DE
004     PUSH HL
005     DI
006     CALL 4008H
007     CALL 4035H
008     OR A
009     JP NZ,400EH
010     CALL 400BH
011     EI
012     POP HL
013     POP DE
014     POP BC
015     POP AF
016     RET
```

BASIC PROGRAM LISTING - READ ONLY

```
5 IY=PEEK(30897)+256*PEEK(30898)+1-65536
6 DIM B(128): POKE 30862,0: POKE30863,114
8 GOSUB 1000
10 CLS:PRINT"    SECTOR READ/WRITE UTILITY"
20 PRINT:PRINT:INPUT"READ OR WRITE SECTOR (R/W)";Z$
30 IF Z$="R" THEN POKE 29193,53 ELSE POKE 29193,50
40 PRINT:INPUT"TRACK #";T: INPUT"SECTOR #";S
50 POKE IY+18,T: POKE IY+17,S
60 IF Z$="W" THEN
70 X=USR(0)
80 BUFF=PEEK(IY+49)+256*PEEK(IY+50)
90 FOR Z=1 TO 128
100 B(Z)=PEEK(Z+BUFF-65537)
110 NEXT Z
120 CLS: FOR Z= 1 TO 128
130 H=INT(B(Z)/16)
140 IF H>9 THEN H$=CHR$(55+H) ELSE H$=CHR$(48+H)
150 L=B(Z)-H*16
160 IF L>9 THEN H$=H$+CHR$(55+L) ELSE H$=H$+CHR$(48+L)
165 P=INT((Z-1)/8) :PRINT@ P*32+(Z-P*8-1)*3,H$
167 IF B(Z)>31 AND B(Z)<123 THEN H$=CHR$(B(Z)) ELSE H$="."
170 PRINT@ P*24+23+Z,H$
180 NEXT Z
190 GOTO 190
```

```
1000 FOR Z=29184 TO 29207: READ A: POKE Z,A: NEXT : RETURN
1010 DATA 245,197,213,229,243,205,8,64,205,53,64,183,194,14,64
1020 DATA 205,11,64,251,225,209,193,241,201
```

We now want to add to this program to allow us to write to any sector on the disk. Recall that the machine code was POKEd into screen RAM from DATA in the BASIC program. The BASIC program then modified line 007 of the machine code so that it either CALLed the READ SECTOR or WRITE SECTOR routine, depending on the users response to the INPUT statement. It then asked for the particular track and sector number to be read or written to and POKEd these values into the relevant vectors in the DOS scratchpad. A jump to line 500 was carried out if the user wanted to write to a sector, but no line 500 existed until now.

Firstly, we must decide on what data is going to be written to the sector. We have been using the B array as temporary storage for the 128 bytes to be read or written to a sector, so let's fill this with say, 255's (FFH's). The listing below shows the lines to be added to the BASIC program for writing sectors. Line 500 fills the B array with 255's. Line 510 calculates the location of the data buffer, in the DOS scratchpad, used by the DOS for transferring data to and from sectors. Line 520 copies the contents of the B array into this buffer. The machine code routine is called at line 530. Line 540 is simply to prevent the program from executing the routine at line 1000.

ADD-ON CODE FOR WRITING TO SECTOR

```
500 FOR Z=1 TO 128: B(Z)=255: NEXT Z
510 BUFF=PEEK(IY+49)+256*PEEK(IY+50)
520 FOR Z=1 TO 128: POKE Z+BUFF-65537,B(Z): NEXT Z
530 X=USR(0)
540 STOP
```

Type in the extra lines listed above. Save your program again, then insert a freshly initialised disk. Run the program, selecting 'W' for write sector, and track 0 sector 0. The disk should spin for a short time, after a short delay. When the program breaks, re-run it, this time selecting 'R' for read sector, track 0 sector 0. You should see the screen fill with FF's, indicating that the sector has successfully been filled with FF bytes. Now break the program and try a DIR. You should recall that track 0 sector 0 is the first sector in the directory, so when the directory appears it will be full of FF's (inverse ?'s on my VZ200)!

You can experiment yourself. Try reading one of the sectors in the directory (track 0, sector 0-14) into the B array. By only changing the characters in the filetypes and filenames of the entries, you can create some interesting effects. You may like to firstly create a file which doesn't contain anything, and use its

filename to display graphics characters or the like. Create a dummy file on a freshly initialised disk. Just type SAVE"DUMMY" with no program in memory. Now, using the program above, with line 500 changed to just a REM statement, read track 0 sector 0. Break the program, and enter the following command directly:

```
B(3)=31:FOR Z=4 TO 10:B(Z)=28:NEXT Z:GOTO 10
```

Select write to track 0 sector 0. Now, when the disk is DIRed, the screen will clear before the directory appears (the codes 31 and 28 we have placed in the filename are 'clear screen' and 'home' respectively). Experiment further yourself!

In the above example, we've been using only two of the routines in the DOS itself; namely the Read Sector routine, and the Write Sector routine. These were called by the machine code program. There are many routines in the DOS which the user can make use of The Technical Reference Manual gives a list of most of these. The list below is a little more comprehensive, however.

DOS ROUTINES - ENTRY POINTS

Jump Table	Start Address Version 1.2A	Start Address Version 1.2B	Description
4008H	5F41H	5F1BH	Disk Power On
400BH	5F52H	5F2CH	Disk Power Off
400EH	4241H	4241H	Error Handling
4011H	4717H	4717H	Read track map
4014H	4749H	4749H	Clear a sector
4017H	4754H	4754H	Save track map
401AH	4B08H	4B04H	Initialise disk
401DH	5367H	535FH	Command Interp.
4020H	53B9H	53B1H	ASCII to Hex
4023H	53EAH	53E2H	Read IDAM
4026H	587BH	5873H	Create entry
4029H	58BFH	58B7H	Find empty sector
402CH	5913H	590BH	Find entry in dir
402FH	5968H	595CH	Find empty in dir
4032H	59A1H	5991H	Write Sector
4035H	5B27H	5B17H	Read Sector
4038H	5EBEH	5E98H	Delay C msec
403BH	5ECEH	5EA8H	Step in
403EH	5F01H	5EDBH	Step out
4041H	43B1H	43B1H	Load a file
4044H	446EH	446EH	Save a file

DOS Command Entry Points:

4A80H	4A7CH	REN Command
4B08H	4B04H	INIT Command
4D78H	4D74H	DRIVE Command
4D92H	4D8EH	IN# Command
4E64H	4E5CH	PR# Command
4FFBH	4FF3H	DCOPY Command
45DBH	45DBH	RUN Command
48C4H	48C4H	BLOAD Command

48EFH	43EFH	BRUN Command

In the first table, three addresses are given for each routine. The first is the routines addresses in the jump table. This is a list of jumps, found at the start of the DOS, which simply jump to the routines themselves. This table has been used, so that when new versions of the DOS are produced, and the actual addresses of the routines change, the jump table is simply updated with the new address. For this reason, programmers should always CALL the address found in the jump table when using these routines to ensure compatibility with all VZ's.

The next two columns show the actual starting addresses of the routines in memory for two different DOS versions I am aware of. These are called version 1.2A and 1.2B (both display V1.2 at start up). Other older versions of DOS are about, so be careful when using these addresses (you should only refer to them for the purpose of examining the routines, not when calling them!). The last column describes each function in brief.

The second list is a list of where the actual command routines themselves reside in memory. These are not found in the jump table, so be very careful when calling them as your DOS may vary.

Of course each routine has special entry and exit conditions; some of these are listed in the Technical Manual, but I have found this to be unreliable,

---

A PLEA FOR HELP

DEAR PETER,
THANKYOU FOR YOUR PROMPT REPLY TO MY LETTER ASKING FOR A WORD PROCESSING TAPE. I HAVE TRIED TO RUN IT AND IT WORKS FINE UNTIL I TRY TO PRINT IT TO THE PRINTER OR SCREEN. I THEN GET A SERIES OF PRISON ARROWS IN INVERSE. AND THEY ARE ALL POINTING TO THE LEFT. IF I TRY TO MOVE THE CURSOR UP OR DOWN IT DOES THE SAME.

I WILL LLIST THIS FIRST AND THEN RUN TO SHOW YOU WHY I NEED YOUR W.P. TAPE.

YOU WILL SEE ON THE REVERSE SIDE WHAT I GOT WHEN I TRIED TO PRINT TO THE PRINTER.
I AM USING AN EPSOM LX800 PRINTER DRIVEN BY A VZ300 AND A PRINTER INTERFACE.

I HAVE AN APPLE 11E COMPUTER WHICH WON'T TALK TO THE PRINTER. THIS UNIT HAS A DUAL DISK DRIVE WHICH IS IN ITSELF A SEPERATE UNIT FROM THE REST.
QUESTION. COULD I USE THIS UNIT WITH MY VZ300 USING A DISK CONTROLLER?

YOURS SINCERELY,

LOU ELLMERS.

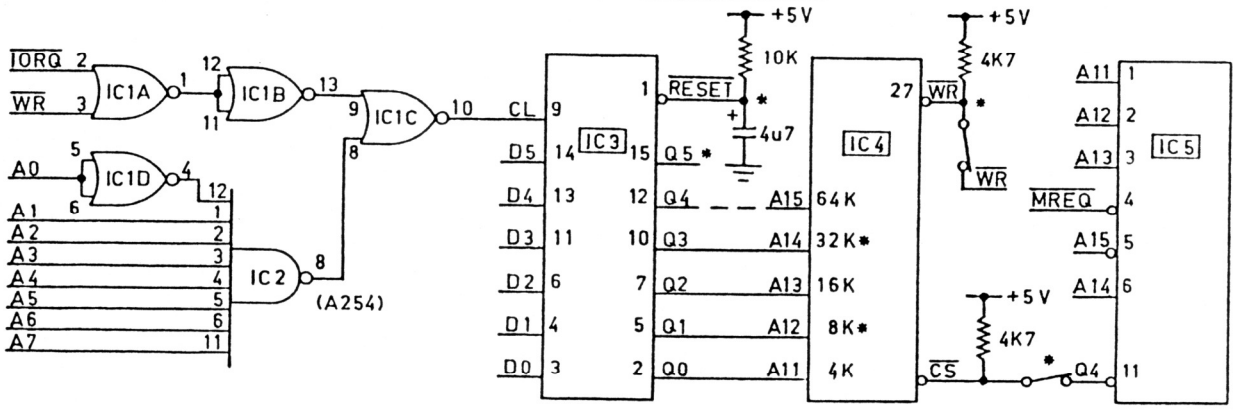
P.O.BOX 1
PAIHIA
6/10/89

Lou Ellmers

4K-64K RAM/EPROM BY J. LEON

4K-64K SIDEWAYS RAM/EPROM FOR 6000H-67FFH RANGE BY JOE LEON - AUGUST 1988

ORIGINAL BY DAVE MITCHELL - 1987



IC1-74LS02 IC2-74LS30 IC3-74LS174 IC4-SEE TEXT IC5-74LS138 & SEE TEXT

The original idea for sideways eprom came from Dave Mitchell and he has had up and running a 4K eprom at 6000H-67FFH, the 2K gap left by the DOS in the cartridge area. It is switched in 2K banks as required. I took the idea to it's logical conclusion by making provision for RAM up to 32K and EPROM up to 64K. That means it's possible to have 32 X 2K banks of memory at your disposal, all banks occupying the same 2K memory location.

In practice I can't see more than 8K of RAM or EPROM being used. Any routines in the 2K gap do slow basic programs down, but it's a small price to pay for added functions available at power up like Dave's Extended Dos which I would'nt be without. And now to the circuit :-

IC-1 and IC-2 together form a decoder for address 254 and the output at pin 10 of IC-1C is used to trigger IC-3.

IC-3 is a latch which is set by the previous stage. The latch is used to select the required bank by switching the relevant RAM/EPROM address lines HI or LO. The Resistor and Capacitor at pin 1 of IC-3 perform a POWER ON RESET functions setting all output latches LO selecting bank (0) at power up.

Pin 1 of IC-3 can be connected to pin 2 of memory expansion connector and in that case the Resistor/Capacitor at pin 1 is no longer required.

IC-5 decodes the range 6000H-67FFH and is the CHIP SELECT for RAM/EPROM.

IC-4 - This needs a little explaining as it can be an 8K or 32K RAM or 4K to 64K EPROM. The pinouts differ on most of them and for that reason are not shown. In case of RAM the Resistor and switch at pin 27 is required and performs WRITE/WRITE PROTECT functions and not required for EPROM.

The CS (CHIP SELECT SWITCH) is used to ENABLE/DISABLE the RAM/EPROM which can be desirable and necessary with the cartridge word processor as it wont work with any RAM/EPROM active at 6000H-67FFH.

Connect all DATA lines to IC-3 and outputs not required can be left unconnected. As only a few IC'S are involved unit can be constructed on VEROBOARD. In my case I mounted the 8K RAM on David Newcombe's ram board and piggy backed the 4 control IC'S using VEROBOARD.

To use is simplicity itself, just load your routines from tape or disk and use the following command to select desired bank :-
OUT 254,N (N=0 to 3 for 8K RAM/EPROM)

STANDARD MICROSOFT / MACHINE	ABS Gives absolute value of expression.	ASC Returns ASCII value of first character of string.	ATN Arctangent of expression.	AUTO	CALL Calls assembler language sub-routine.	CHAIN Call a new program & pass variables to it.	CHR$ Gives one-char-string with ASCII code of exp.	CLEAR Clear selected vaariables.	CLOSE Closes disk files —closes all files if no specification.	CONT Continue program execution.	COS Cosine of expression.
	ABS(exp)	ASC(string)	ATN(exp)	AUTO [lineno, val]	CALL var[(var. var . . .)]	CHAIN "filename"	CHRS(exp)	CLEAR[exp. exp]		CONT	COS(exp)
APPLESOFT	ABS(exp)	ASC(string)	ATN(exp)		CALL addr	CHAIN "filename"	CHRS(exp)	CLEAR	CLOSE "filename"	CONT	COS(exp)
ATARI	ABS(exp)	ASC(string)	ATN(exp)			RUN"C:" NB: program must have been saved using SAVE "C"	CHRS	CLR	CLOSE [#fileno. fileno . . .]	CONT	COS(exp)
BBC MICRO	ABS(exp)	ASC(string)	ATN(exp)	AUTO [lineno. val]	CALL addr [.var,var . . .]	CHAIN "filename"	CHRS(exp)	CLEAR	CLOSE #fileno Note:CLOSE #0 to close all files		COS(exp)
COMMODORE 64	ABS(exp)	ASC(string)	ATN(exp)		SYS(addr)		CHRS(exp)	CLR(exp)	CLOSE fileno	CONT	COS(exp)
MICROBEE	ABS(real-exp)	ASC(string)	ATAN(real-exp)	AUTO (lineno. val]			CHR(integer-exp)	STRS(int-exp) Note: set limits for string memory		CONT	COS(real-exp)
PET	ABS(exp)	ASC(string)	ATN(exp)		SYS(addr)		CHRS(exp)	CLR	CLOSE fileno	CONT	COS(exp)
TRS-80/SYSTEM 80	ABS(exp)	ASC(string)	ATN(exp)	AUTO [lineno. val]			CHRS(exp)	CLEAR(exp) Note: Clears string space if exp given	[depends on OS: consult OS manual]	CONT	COS(exp)
VIC-20	ABS(exp)	ASC(string)	ATN(exp)		SYS addr		CHRS(exp)	CLR	CLOSE #fileno	CONT	COS(exp)
VZ200	ABS(exp)	ASC(string)	ATN(exp)				CHRS(exp)	CLEAR[exp] N clears string space		CONT	COS(exp)
ZX81	ABS(exp)	CODE(string) Note: ZX81 does not use ASCII code	ATN(exp)		LET var = USR (addr) Note: equivalent statement		CHRS(exp) Note: ZX81 does not use ASCII code	CLEAR	N/A — ZX81 does not support file-handling	CONT	COS(exp)
ZX SPECTRUM	ABS(exp)	CODE(string)	ATN(exp)		LET var = USR (addr) Note: roughly equivalent		CHRS(exp)	CLEAR	Consult Microdrive manual	CONTINUE	COS(exp)